CLAIMS

1. A method comprising:

receiving an original digital good; and

randomly applying various forms of protection to the original digital good to produce a protected digital good.

2. A method as recited in claim 1, wherein the randomly applying comprises pseudo randomly applying the various forms of protection according to pseudo random techniques.

3. A method as recited in claim 1, wherein the applying comprises randomly selecting the forms of protection from a set of available forms of protection.

4. A method as recited in claim 1, wherein the applying comprises applying the various forms of protection to randomly selected portions of the original digital good.

5. A method as recited in claim 1, wherein the various forms of protection are selected from a group of protection tools comprising code integrity verification, acyclic code integrity verification, cyclic code integrity verification, secret key scattering, obfuscated function execution, encryption/decryption, probabilistic checking, Boolean check obfuscation, in-lining, reseeding pseudo random number generators with time varying inputs, anti-disassembly methods, varying execution paths between runs, anti-debugging methods, and time/space separation between tamper detection and response.

6. A method as recited in claim 1, wherein the applying comprises applying a form of protection in which a checksum can be computed on a set of bytes of the digital good without actually reading the bytes.

7. A computer-readable medium comprising computer-readable instructions that, when executed by a processor, direct a computer system to perform the method as recited in claim 1.

8. A method comprising:

segmenting a digital good into a plurality of segments;

selecting multiple segments from the plurality of segments; and

transforming the selected segments according to different protection techniques to produce a protected digital good having a composite of variously protected segments.

9. A method as recited in claim 8, wherein at least two of the segments overlap one another.

10. A method as recited in claim 8, wherein the selecting comprises randomly selecting the segments.

11. A method as recited in claim 8, wherein the transforming comprises transforming the selected segments according to randomly chosen protection techniques.

12. A method as recited in claim 8, wherein the transforming comprises:

augmenting at least one segment using a certain protection technique; and

inserting a checkpoint, which may be used to evaluate a validity of the augmented segment, within the protected digital good but outside of the augmented segment being evaluated.

13. A method as recited in claim 8, further comprising receiving quantitative parameters indicative of how much the protected digital good should be altered.

14. A method as recited in claim 13, wherein the transforming is performed to satisfy the quantitative parameters.

15. A method as recited in claim 8, wherein the protection techniques are selected from a group of protection tools comprising code integrity verification, acyclic code integrity verification, cyclic code integrity verification, secret key scattering, obfuscated function execution, encryption/decryption, probabilistic checking, Boolean check obfuscation, in-lining, reseeding pseudo random number generators with time varying inputs, anti-disassembly methods, varying execution paths between runs, anti-debugging methods, and time/space separation between tamper detection and response.

16. A method as recited in claim 8, wherein the transforming comprises applying a protection technique in which a checksum can be computed on a set of bytes of the digital good without actually reading the bytes.

17. A computer-readable medium comprising computer-readable instructions that, when executed by a processor, direct a computer system to perform the method as recited in claim 8.

18. A method comprising:

establishing parameters prescribing a desired quantity of protection to be applied to a software product;

parsing the software product into code sections;

selecting at least one code section;

augmenting the selected code section to add protection qualities; and

repeating the selecting and the augmenting for different code sections until the desired quantity of protection has been applied.

19. A method as recited in claim 18, wherein the establishing comprises enabling a user to enter the parameters.

20. A method as recited in claim 18, wherein the augmenting comprises applying a protection technique selected from a group of protection techniques comprising code integrity verification, acyclic code integrity verification, cyclic code integrity verification, secret key scattering, obfuscated function execution, encryption/decryption, probabilistic checking, Boolean check obfuscation, in-lining, reseeding pseudo random number generators with time varying inputs, anti-disassembly methods, varying execution paths between runs, anti-debugging methods, and time/space separation between tamper detection and response.

21. A method as recited in claim 18, wherein the augmenting comprises applying a protection technique in which a checksum can be computed on a set of bytes of the digital good without actually reading the bytes.

22. A computer-readable medium comprising computer-readable instructions that, when executed by a processor, direct a computer system to perform the method as recited in claim 18.

23. A production system, comprising:

a memory to store an original digital good; and

a production server equipped with a set of multiple protection tools that may be used to augment the original digital good for protection purposes, the

production server being configured to parse the original digital good and apply protection tools selected from the set of protection tools to various portions of the original digital good in a random manner to produce a protected digital good having a composite of variously protected portions.

24. A production system as recited in claim 23, wherein the protection tools are selected from a group of protection tools comprising code integrity verification, acyclic code integrity verification, cyclic code integrity verification, secret key scattering, obfuscated function execution, encryption/decryption, probabilistic checking, Boolean check obfuscation, in-lining, reseeding pseudo random number generators with time varying inputs, anti-disassembly methods, varying execution paths between runs, anti-debugging methods, and time/space separation between tamper detection and response.

25. A production system as recited in claim 23, wherein the production server applies a protection tool that enables a checksum to be computed on a set of bytes of the digital good without actually reading the bytes.

26. A production system as recited in claim 23, wherein the production server has a pseudo random generator to introduce randomness into the application of the protection tools to various portions of the original digital good.

27. An obfuscation system, comprising:

a parser to parse a digital good into a plurality of segments;

a set of protection tools that may be applied to the segments of the digital good to augment the segments with protection qualities;

a target segment selector to select at least one segment from the plurality of segments; and

a tool selector to select at least one protection tool from the set of protection tools and apply the selected protection tool to the selected segment.

28. An obfuscation system as recited in claim 27, wherein the protection tools are selected from a group of protection tools comprising code integrity verification, acyclic code integrity verification, cyclic code integrity verification, secret key scattering, obfuscated function execution, encryption/decryption, probabilistic checking, Boolean check obfuscation, in-lining, reseeding pseudo random number generators with time varying inputs, anti-disassembly methods, varying execution paths between runs, anti-debugging methods, and time/space separation between tamper detection and response.

29. An obfuscation system as recited in claim 27, wherein the target segment selector comprises a pseudo random generator to enable random selection of the segment.

30. An obfuscation system as recited in claim 27, wherein the tool selector comprises a pseudo random generator to enable random selection of the protection tool.

31. An obfuscation system as recited in claim 27, further comprising a quantitative unit to specify a quantity of protection qualities to be added to the digital good.

32. A client-server system, comprising:

a production server to randomly apply various forms of protection to a digital good to produce a protected digital good; and

a client to store and execute the protected digital good, the client being configured to evaluate the protected digital good to determine whether the protected digital good has been tampered with.

33. One or more computer-readable media having computer-executable instructions that, when executed, direct a computing device to:

parse a digital good into a plurality of segments; and

apply multiple different protection tools to various segments in a random manner to produce a protected digital good having a composite of variously protected portions.

34. One or more computer-readable media as recited in claim 33, further comprising computer-executable instructions to randomly select the protection tools from a set of available protection tools.

35. One or more computer-readable media as recited in claim 33, further comprising computer-executable instructions to apply the protection tools to randomly selected portions of the original digital good.

36. One or more computer-readable media as recited in claim 33, wherein the protection tools are selected from a group of protection tools comprising code integrity verification, acyclic code integrity verification, cyclic code integrity verification, secret key scattering, obfuscated function execution, encryption/decryption, probabilistic checking, Boolean check obfuscation, in-lining, reseeding pseudo random number generators with time varying inputs, anti-disassembly methods, varying execution paths between runs, anti-debugging methods, and time/space separation between tamper detection and response.